ARMY RESEARCH LABORATORY

OpenGL Performance Evaluation on Multiple Computer Platforms

Sean Ho

ARL-TR-2374

November 2001

Approved for public release; distribution unlimited.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Adelphi, MD 20783-1197

ARL-TR-2374 November 2001

OpenGL Performance Evaluation on Multiple Computer Platforms

Sean Ho

Computational and Information Sciences Directorate

Abstract

With the recent advent of 3D graphics hardware for personal computer (PC), it is worthwhile to exploit the cost effectiveness and OpenGL performance issues among currently available commercial off-the-self (COTS) computers. Graphics hardware vendors typically list several gross measurements of system performance when releasing new graphics hardware. Often these coarse or subjective figures do not represent how a software application performs. On the other hand, one seldom sees the same benchmark performed on machines across multiple platforms and operating systems, i.e., Intel-based PCs and RISC-based UNIX workstations. This document reports the results obtained from running two OpenGL benchmark programs, SPECviewperf 6.1.2 and SPECglperf 3.1.2, on existing computer workstations at ARL.

Contents

Figures

Tables

1. Introduction

Virtual Geographic Information System (VGIS) [1] is a geographic information 3-D visualization system developed in-house on top of OpenGL. Since it does not rely on any commercial package, it is portable to any computer platform with OpenGL [2] support. Consequently, it is worthwhile to exploit the cost-effectiveness and OpenGL-performance issues among currently available commercial off-the-shelf (COTS) computers. Graphics hardware vendors typically list several gross measurements of system performance when releasing new graphics hardware. Often these coarse or subjective figures do not represent how an application performs. On the other hand, one seldom sees the same benchmark performed on machines across multiple platforms and operating systems, i.e., Intel-based PCs and RISC-based UNIX workstations. This report presents the results obtained from running two OpenGL benchmark programs, SPECviewperf 6.1.2 [3] and SPECglperf 3.1.2 [4] on existing computer workstations at ARL. These include an SGI Onyx InfiniteReality, an SGI Octane MXE workstation, and a Micron Intel processor-based PC with a GeForce 256 graphics card. Both Windows 98 SE and RedHat 7.0 are installed and benchmarked on the same PC. Table 1 shows the hardware specifications on the three target systems:

Table 1. This table shows the hardware specifications on the three target systems.

	SGI Onyx IR	SGI Octane MXE	Micron PC
CPU	194-MHz MIPS R10000	250-MHz MIPS R10000	800-MHz Intel Pentium III
CPU count	4	2	1
Memory size	2048 MB	896 MB	256 MB
Operating System	IRIX 6.5	IRIX 6.5	Windows 98 SE/Linux 2.2.16-22
OpenGL Vendor	SGI	SGI	NVIDIA Corporation
OpenGL Version	1.2	1.2	1.2.1
OpenGL Renderer Driver	IRS/S/1(RM6)/ 64(MB)/4(GE) SGI	IMPACT 2(GE)/2(RE)/4(MB) SGI	GeForce 256/APG/DDR(32MB) NVIDIA Detonator 3/Xfree86 4.0.1 build 0.9.5
Display resolution	1600 × 1200 32-bit color depth	1600 × 1024 32-bit color depth	1600 × 1200 32-bit color depth

2. SPECviewperf and SPECglperf

2.1 Common Background

SPECviewperf and SPECglperf are portable OpenGL performance benchmark programs written in C. Both were developed by the OpenGL Performance Characterization (OPC) group of the Standard Performance Evaluation Corporation (SPECopc) [5]. The goal of the SPECopc project is to provide unambiguous, vendor-neutral measures for comparing the performance of OpenGL implementations across vendor platforms, operating systems, and windowing environments. The SPECopc project group maintains a single source code version of the SPECviewperf and SPECglperf code. The sources were downloaded, compiled, and linked on the target Onyx, Octane, and Linux systems. The Windows version of the benchmarks was installed via Microsoft's InstallShield.

2.2 Some Differences

Even though both benchmarks measure the graphics performance of a computer system through the OpenGL Applications Programming Interface (API), they were designed with different goals in mind. SPECviewperf draws models with different sizes of primitives as one would see in an actual application. On the other hand, SPECglperf artificially assigns a specific size to every primitive drawn within a test. SPECviewperf emulates what an application would do graphically and measures it; SPECglperf makes no such attempt. Instead, SPECglperf measures the highest performance or upper bound of the target system in a more controlled environment.

SPECviewperf reports result in frames drawn per second (FPS), whereas SPECglperf reports in primitives drawn per second. As an analogy, SPECglperf is like a speedometer measuring top speed, while SPECviewperf would be a stopwatch measuring the average speed through a slalom course. For this report, data collected from running the two benchmarks will be presented. A brief description of the test and a conclusion will precede and follow the results, respectively. For both benchmarks, raw data are omitted to shorten the report and results are presented in graphical form for ease of comparison.

3. SPECviewperf

SPECviewperf is a real-world benchmark in the sense that it is comprised of the OpenGL rendering portion of independent software vendor (ISV) applications. It consists of six viewsets. A viewset is a group of individual runs of SPECviewperf that attempts to characterize the graphics-rendering portion of an ISV's application. The SPECopc project group does not develop these applications, but instead they are provided by the ISVs themselves. A brief description of each viewset will be presented. A more detailed description on each viewset and its individual test cases can be found on the following website: http://www.spec.org/gpc.

3.1 Test Procedures

Source codes were compiled and linked with the most up-to-date OpenGL library on the Onyx, Octane, and Linux systems. Only essential tasks as required by the OS were running during the test. At the end of each test run within a viewset, an image was captured in portable network graphics (PNG) format [6] for the purposes of visual quality assessment and verification. The PNG format uses loss less compression and supports up to 16 bits per color component.

3.2 Results

3.2.1 Awardvs-04

This viewset is extracted from Alias/Wavefront's Advanced Visualizer software (Awadvs-04). It tests the animation of a 3-D model with varying shading methods, e.g., material, smooth, and flat. All operations within this viewset are performed in immediate mode with double-buffered windows

As can seen from figure 1, the GeForce 256-based system outperforms the SGIs by at least 30 percent. The Linux system performs better than its Windows counterpart in this test.

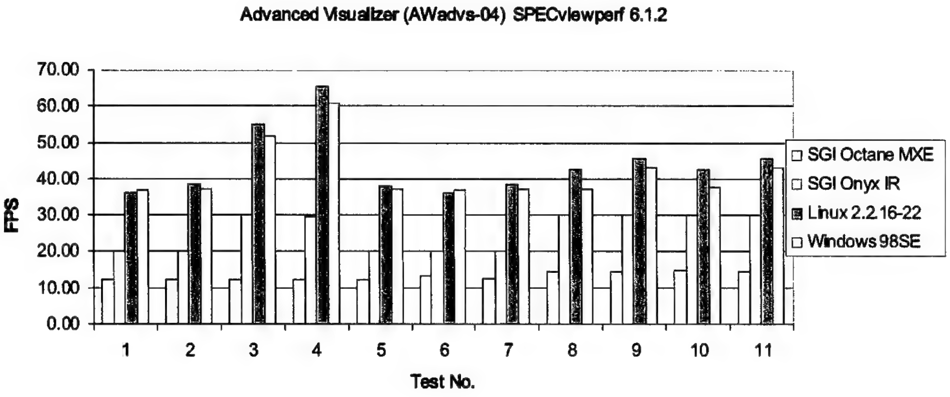


Figure 1. This graph shows the performance of the four target systems from running the Awadvs-04 benchmark in SPECviewperf 6.1.2.

3.2.2 DRV-07

DesignReview, provided by Intergraph Corporation, is a 3-D computer model review package specifically tailored for plant design models. The shaded model used here contains 367,178 vertices in 42,821 primitives. The wire frame model contains 1,599,755 vertices in 94,275 primitives.

As can be seen from figure 2, the GeForce 256-based PC system's frame rate doubles that of the SGI Onyx IR.

3.2.3 DX-06

The IBM Visualization Data Explorer (DX) is a general-purpose software package for scientific data visualization and analysis. These tests visualize a set of particle traces through a vector flow field. The object represented in the test has about 3000 triangle meshes containing approximately 100 vertices each. All tests assume Z buffering with one light source in addition to specification of a color at every vertex. Triangle meshes are the primary primitives for this viewset.

The GeForce 256 performed better than the SGIs in 9 of the 10 tests as shown in figure 3. The only case in which the Onyx outperforms the GeForce 256 is in test 8 where the model is rendered in triangle meshes with two-sided lighting.

3.2.4 Light-04

The Lightscape Visualization System from Discreet Logic Incorporated uses a progressive refinement radiosity algorithm to produce useful visual re-

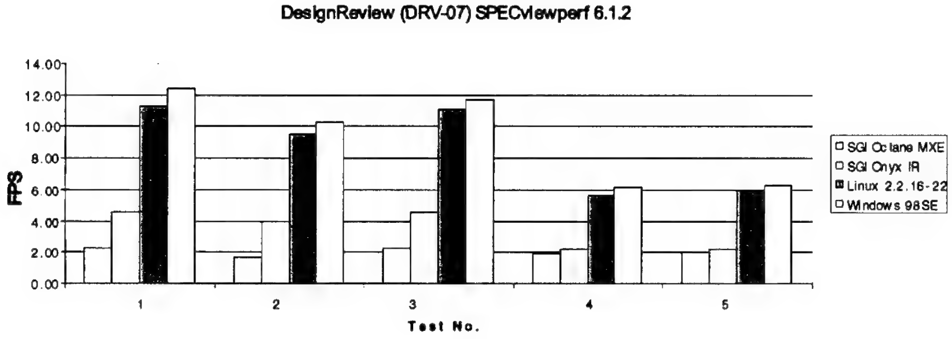


Figure 2. This graph shows the performance of the four target systems from running the DRV-07 benchmark in SPECviewperf 6.1.2.

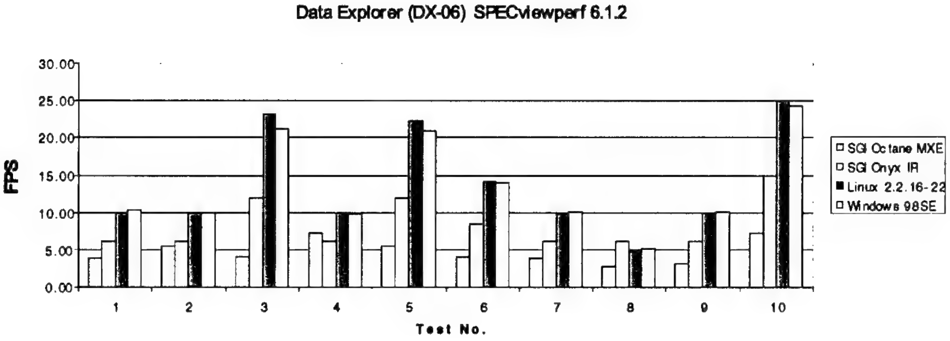


Figure 3. This graph shows the performance of the four target systems from running the DX-06 benchmark in SPECviewperf 6.1.2.

sults almost immediately upon processing. The quality of the visualization improves as the process continues. Performances in full-screen, solid, and wire-frame walkthroughs of the parliament-building model are recorded. Figure 4 clearly shows that the GeForce 256 system outperforms the SGIs consistently by 45 percent.

3.2.5 MedMCAD-01

Unlike other viewsets, the medMCAD-01 viewset is a "generic" viewset, i.e., it is a representative of a class of applications rather than a single application. The medMCAD-01 viewset is intended to model the graphics performance of a range of medium-scale, immediate-mode, Mechanical Computer Aided Design (MCAD) applications such as Pro/ENGINEER™ from Parametric Technology Corporation (PTC) and SolidWorks from SolidWorks Corporation. The viewset consists of 12 tests, each representing a different mode of operation. Four of the tests use a wire frame model; the other eight use a shaded model. All tests use immediate mode and vertex arrays (glDrawArrays). Each test has two runs: (a) with orthographic projection, and (b) with zoom, and pan (walkthrough) in perspective projection. The shaded model uses 47,000 triangle strips with approximately 444,000 vertices resulting in 349,000 triangles total.

The wire frame model consists of 26,500 line strips, with around 192,000 vertices giving 120,000 lines total. The mean line length is seven pixels. Figure 5 reveals that the GeForce 256 system outperforms the SGIs by at least 50 percent except in two cases, 6 and 10, where a user-defined clipping plane is used.

Figure 4. This graph shows the performance of the four target systems from running the Light-04 benchmark in SPECviewperf 6.1.2.

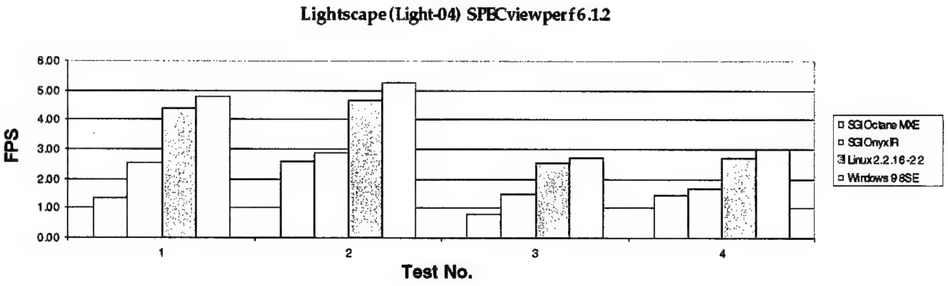
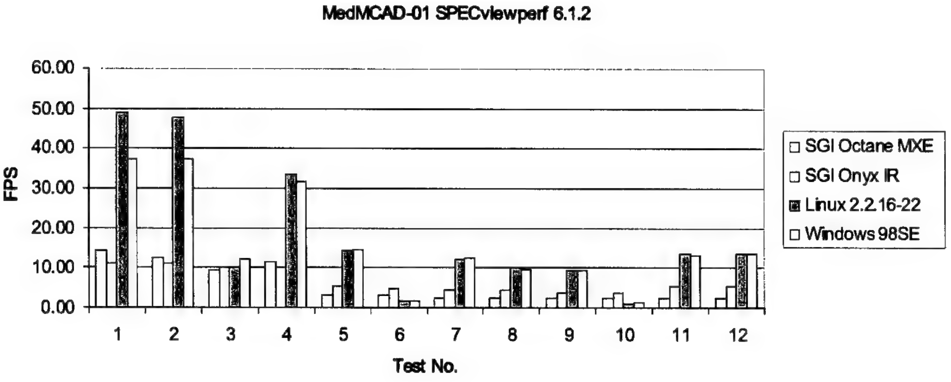


Figure 5. This graph shows the performance of the four target systems from running the MedMCAD-01 benchmark in SPECviewperf 6.1.2.



3.2.6 ProCDRS-03

The ProCDRS-03 is intended to model the graphics performance of PTC's CDRS industrial design software. The viewset consists of 10 tests, each of which represents a different mode of operation within CDRS. The first two tests use a wireframe model, and the remaining 8 use a shaded model. The shaded model is a mixture of triangle strips and independent triangles, with approximately 562,000 vertices in 9300 OpenGL primitives, giving 262,000 triangles total. The wire frame model consists of only line strips, with around 404,000 vertices in 37,000 strips, giving 388,000 lines total. All tests are run in display-list mode. The wireframe tests use antialiased lines since these are the default in CDRS. The shaded tests use one infinite light and two-sided lighting. The texture used in tests 5 through 8 is 512 by 512 pixels in size with 24-bit color.

The GeForce 256 system outperformed the SGIs in all tests except the first two where Onyx led by a wide margin. The first test is a simple wire frame test and the second is a wire frame test with walkthrough. Both tests use antialiased lines which means that the Onyx has hardware antialiasing support whereas the GeForce does not. As figure 6 shows, the Onyx closes the performance gap somewhat in this viewset with textured models in tests 5 through 8.

3.2.7 Weighted Geometric Mean

To derive a composite number for a viewset, each creator of a viewset assigns a weight based on the percentage of time in each path. This composite metric is a derived quantity that is exactly what one would get if one ran the viewset tests for 100 seconds, in which test 1 was run for (100 times weight$_1$) seconds, test 2 for (100 times weight$_2$) seconds, and so on. The WGM formula is $\Pi^{n}_{i=1}$(frames per second$_i$)$^{(w_i)}$, where n is the test number in a viewset. Table 2 and figure 7 represent the calculated WGM for the six viewsets or benchmarks.

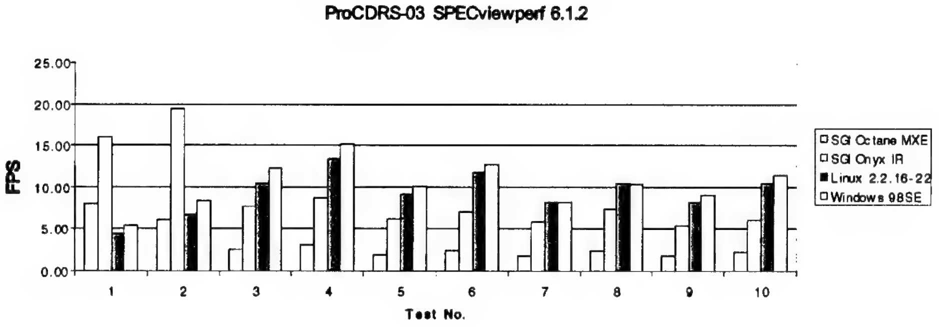
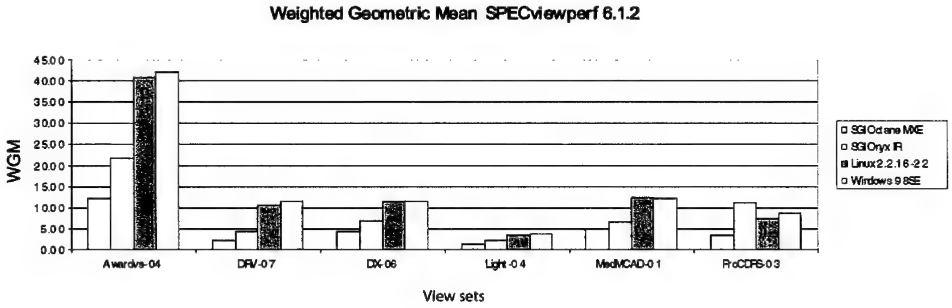


Figure 6. This graph shows the performance of the four target systems from running the ProCDRS-03 benchmark in SPECviewperf 6.1.2.

Table 2. Results in WGM for the six viewsets in SPECviewperf 6.1.2.

	Awardvs-04	DRV-07	DX-06	Light-04	MedMCAD-01	ProCDRS-03
SGI Octane MXE	12.22	2.17	4.47	1.39	4.84	3.32
SGI Onyx IR	21.76	4.27	6.90	2.05	6.50	11.03
Linux 2.2. 16-22	40.95	10.47	11.50	3.44	12.44	7.55
Windows 98SE	42.15	11.42	11.50	3.79	12.22	8.78

Figure 7. This graph shows the plotted WGM results for the six viewsets in SPECviewperf 6.1.2.



3.3 SPECviewperf Conclusion

The Micron PC with the NVIDIA GeForce 256 graphics card running Windows 98 is clearly the winner. From the WGM results, one can see that it outperforms the SGI Octane MXE in every viewset by at least 152 percent and in some cases 426 percent. It also outperforms the SGI Onyx in every viewset by at least 67 percent except the ProCDRS-03 viewset, where the Onyx outperforms it by 26 percent. As mentioned earlier, the GeForce 256 suffers greatly from its lack of line antialiasing hardware support in the last viewset. The successors to GeForce 256, however, do have hardware antialiasing support. After close examinations of all the captured images side by side from each viewset test on the same monitor, I found no visible difference, i.e., the image qualities generated by all systems were compatible.

As expected, the scores revealed little difference in OpenGL performance from the same PC running different operating systems, namely, Windows 98 and Linux. In fact, Windows 98 scored slightly higher than Linux on four of the six viewsets. However, this does not imply that Windows 98 is superior to Linux. This difference is most likely because the NVIDIA's Windows OpenGL driver, the Detonator 3, is more mature than its Linux counterpart, 0.95.

4. SPECglperf

SPECglperf is the second benchmark used to measure the performance of OpenGL 2D and 3D graphics operations. Its operations are performed on low-level primitives (points, lines, triangles, pixels, etc) rather than on entire models such as those used in the SPECviewperf benchmark. A SPECglperf script describes the graphics primitives that will be included in performance tests. Ten RGB scripts are run; their descriptions and results are as follows.

4.1 Results

4.1.1 BgnEnd

This test measures a system's performance in rendering batched primitives between glBegin and glEnd pairs. The number of batched primitives is incremented from 1 to 495.

All lines and line strips are 10 pixels wide. Triangle strips are 25 pixels wide and the quads are 40 pixels wide. The graphs in figures 8 through 15 are generated with varying rendering states and modes.

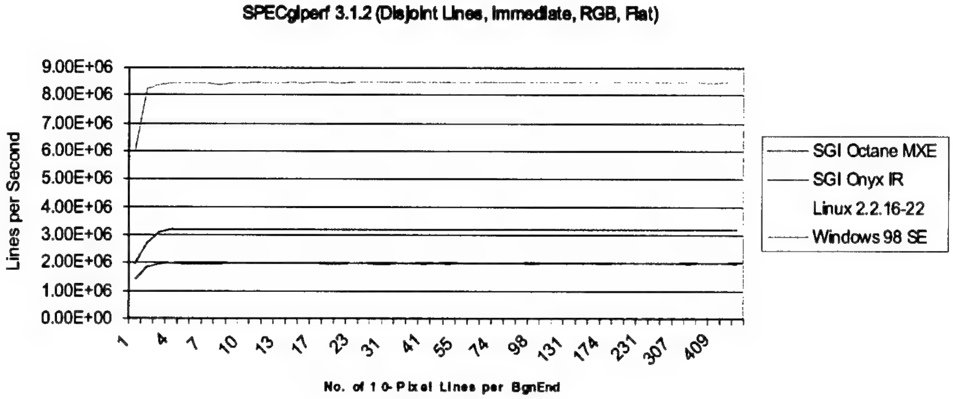


Figure 8. This graph shows the performance of the four target systems on the rendering of disjoint lines in immediate, RGB, and flat-shaded mode.

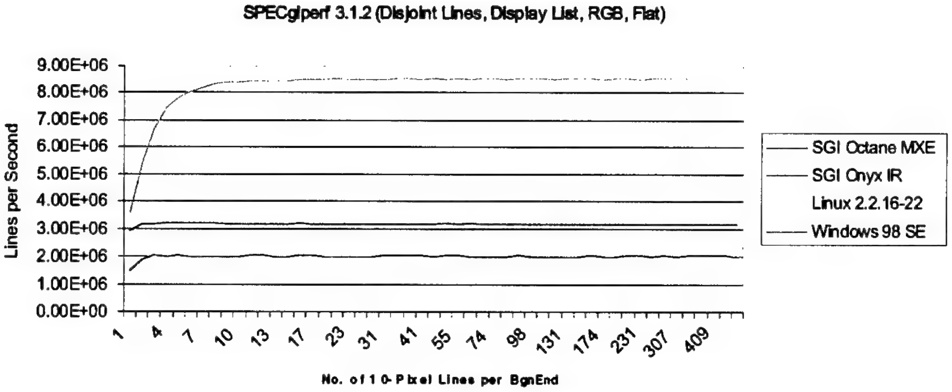


Figure 9. This graph shows the performance of the four target systems on the rendering of disjoint lines in display-list, RGB, and flat-shaded mode.

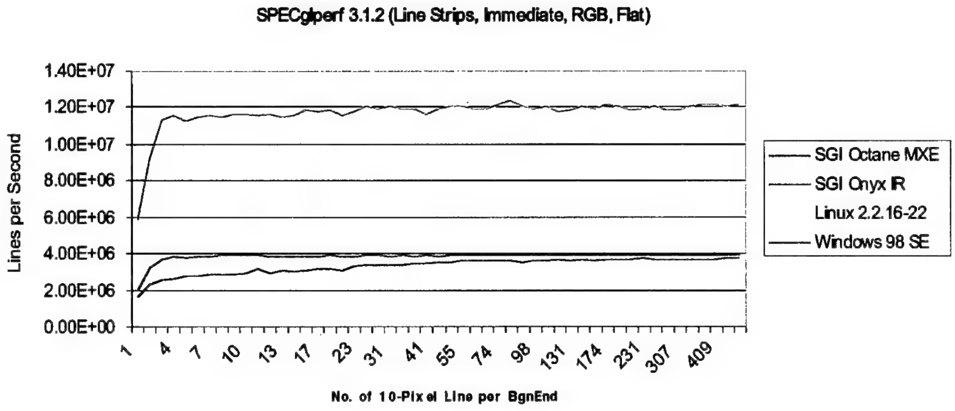


Figure 10. This graph shows the performance of the four target systems on the rendering of lines strips in immediate, RGB, and flat shaded mode.

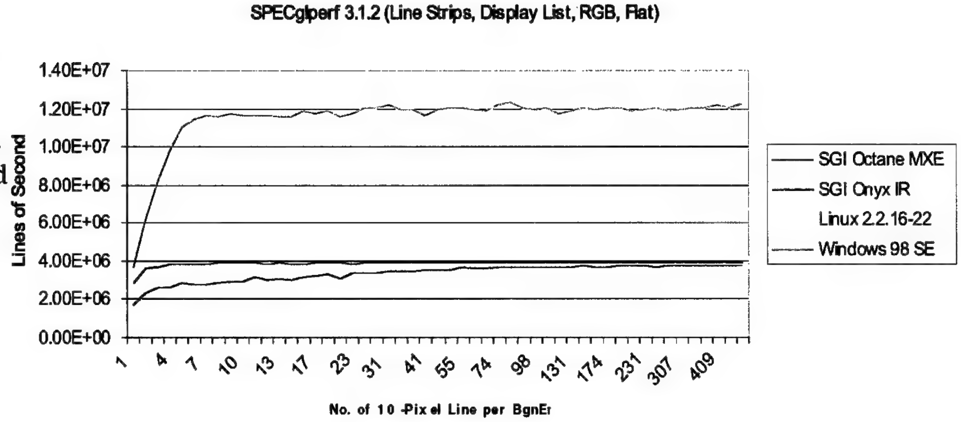


Figure 11. This graph shows the performance of the four target systems on the rendering of lines strips in display-list, RGB, and flat shaded mode.

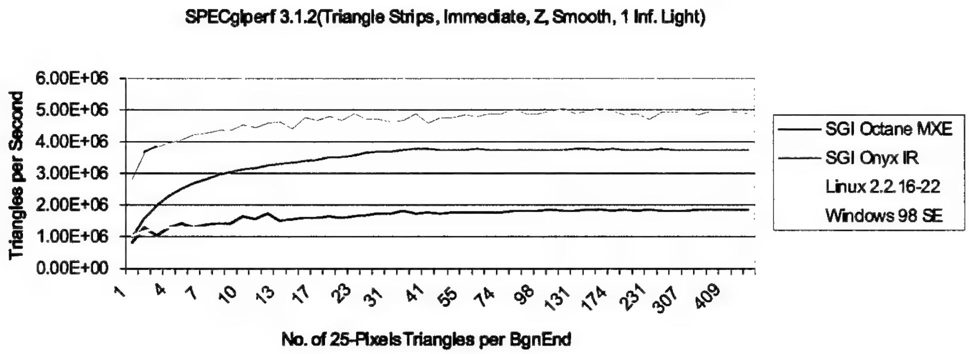


Figure 12. This graph shows the performance of the four target systems on the rendering of triangle strips in immediate, Z buffer, and smooth shaded mode with 1 infinite light source.

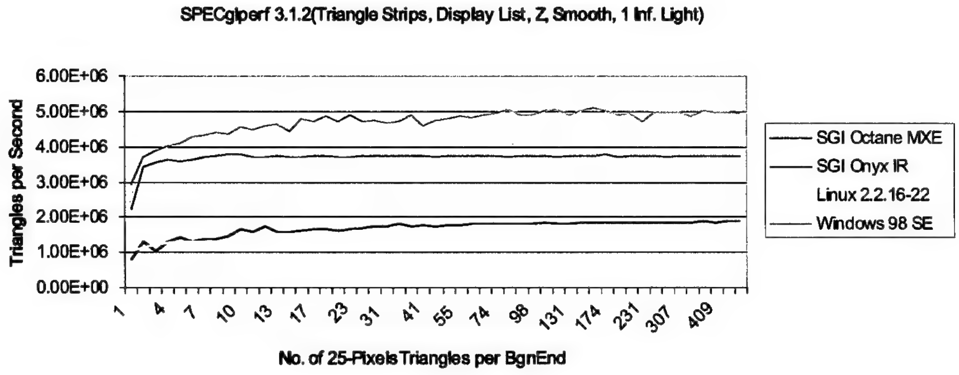


Figure 13. This graph shows the performance of the four target systems on the rendering of triangle strips in display-list, Z buffer, and smooth shaded mode with 1 infinite light source.

The Windows system consistently renders lines and line strips about three times faster than the SGI Onyx system. The gap narrows in the rendering of triangles and quads, however, the Windows system is only about 30 percent faster than the SGI Onyx in triangle rendering, 100 percent faster in rendering of quads in immediate mode and only 20 percent faster in the display-list mode. Surprisingly, the performance of the Linux system is the worst in most of the SPECglperf tests. The driver was installed correctly as shown by the results from the previous SPECviewperf benchmark. The only possible reason is that the 0.95 Xfree 4 Linux driver was not fully implemented to use the GeForce 256's hardware.

Figure 14. This graph shows the performance of the four target systems on the rendering of quads in immediate, Z buffer, and smooth shaded mode with 1 infinite light source.

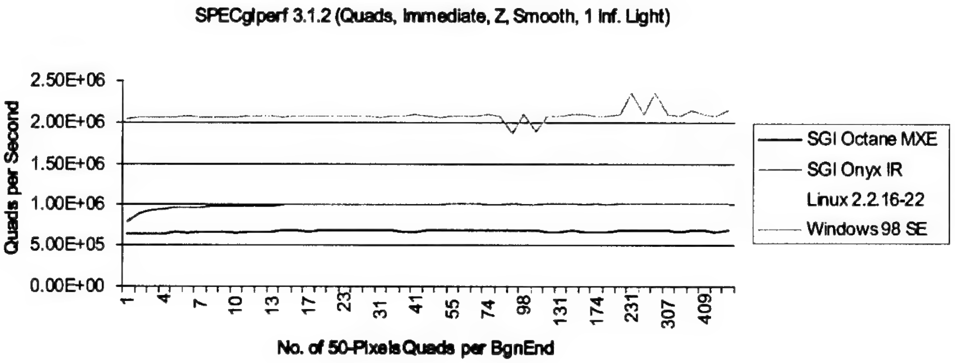
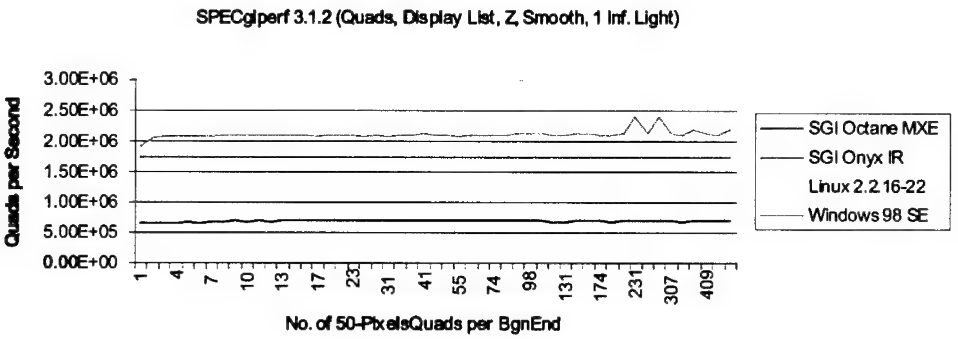


Figure 15. This graph shows the performance of the four target systems on the rendering of quads in display-list, Z buffer, and smooth shaded mode with 1 infinite light source.



4.1.2 CopyPixl (glCopyPixels)

Figure 16 benchmark shows the speed at which the graphics hardware copies various sizes of rectangular pixel arrays from one part of the frame buffer to another. The results are in pixels per seconds.

The GeForce 256 outperformed the SGI Onyx by 800 percent in the 512 × 512 image test. This case is also rare in that the Linux system performed better than its Windows counterpart in all image sizes.

4.1.3 DrawPixl (glDrawPixels)

This script, as presented in figures 17 through 24, determines the rate at which an image is written to the framebuffer in pixels per second.

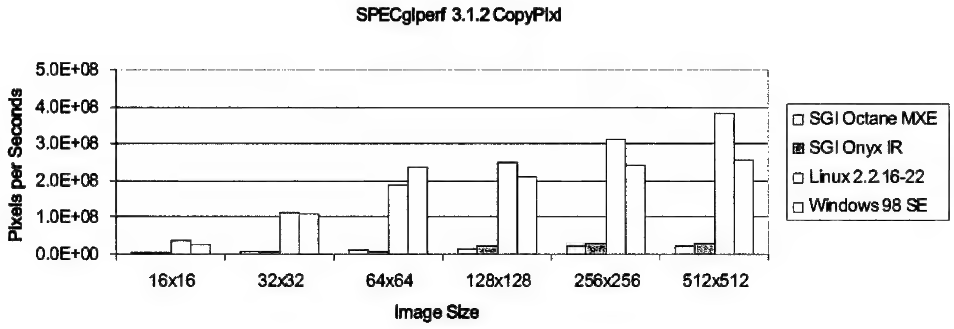


Figure 16. This graph shows the performance of the four target systems on copying pixels of varying image sizes within the frame buffer.

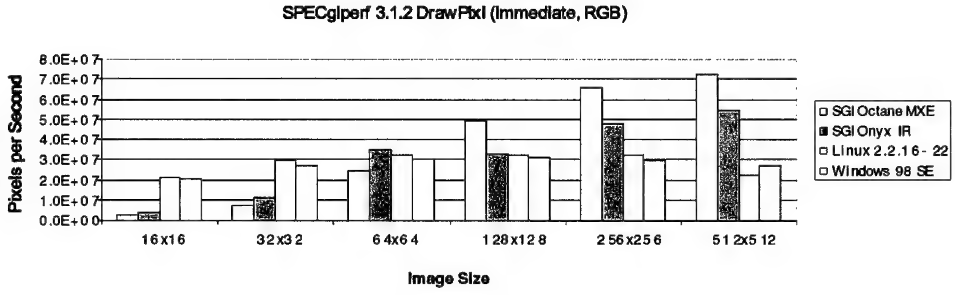


Figure 17. This graph shows the performance of the four target systems on writing pixels of varying image sizes to the framebuffer in immediate and RGB mode.

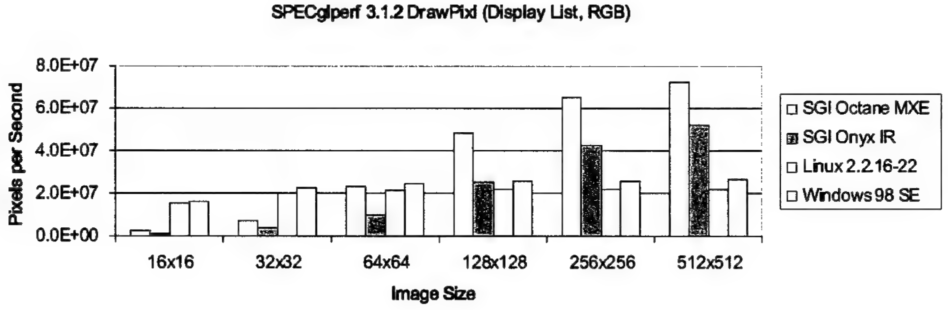


Figure 18. This graph shows the performance of the four target systems on writing pixels of varying image sizes to the framebuffer in display-list and RGB mode.

In the RGB cases in which no zooming is used, the GeForce 256 is faster in writing images with sizes of 32 pixels by32 pixels or smaller. But the SGIs draw large images to the framebuffer faster than the GeForce 256 does. However, in the RGBA mode, the performance gap is not as great between the GeForce and the SGIs in large images. An unexpected result, which is significant, is that the SGI Octane outperformed the Onyx in most of these tests.

Figure 19. This graph shows the performance of the four target systems on writing pixels of varying image sizes to the framebuffer in immediate and RGBA mode.

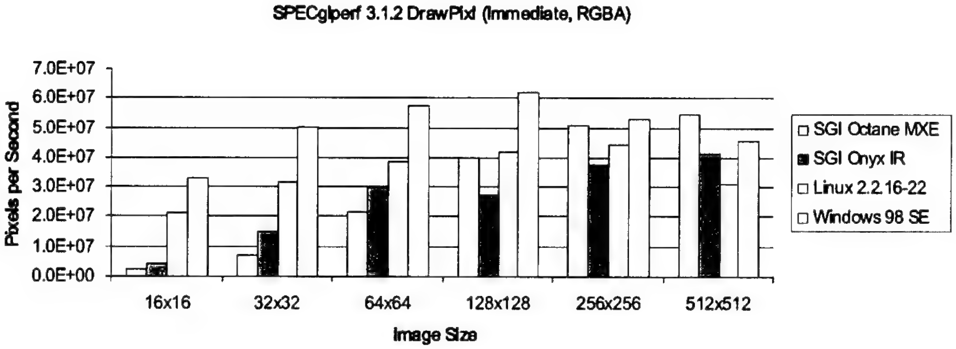


Figure 20. This graph shows the performance of the four target systems on writing pixels of varying image sizes to the framebuffer in display-list, and RGBA mode.

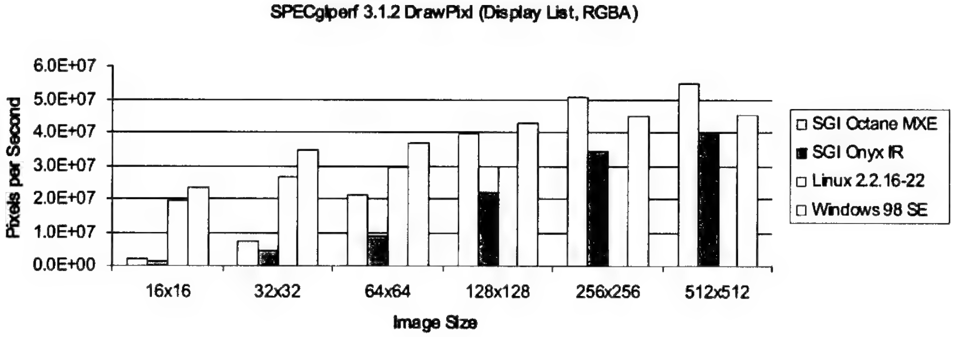
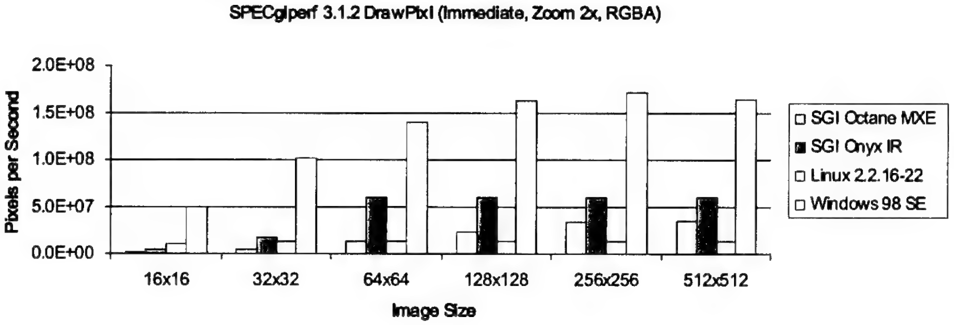


Figure 21. This graph shows the performance of the four target systems on writing pixels of varying image sizes to the framebuffer in immediate, 2x zoom, and RGBA mode.



When zooming (glPixelZoom) is used, the GeForce 256 Windows version outperforms the SGIs across all image sizes by a wide margin. Contrarily, the Linux version was the worst performer in these tests. Once again this indicates that the Linux 0.95 Xfree86-4 OpenGL driver from NVIDIA is not complete.

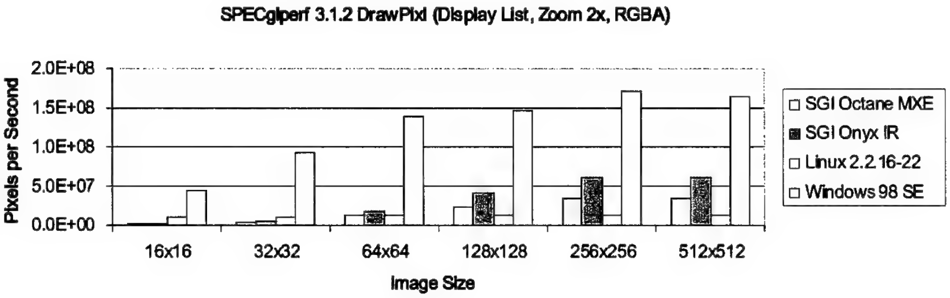


Figure 22. This graph shows the performance of the four target systems on writing pixels of varying image sizes to the framebuffer in display-list, 2x zoom, and RGBA mode.

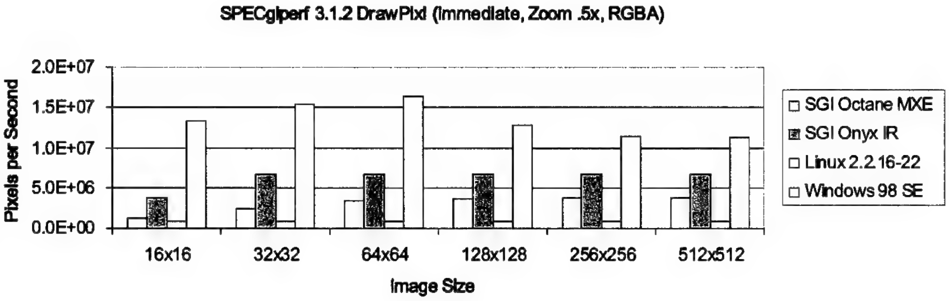


Figure 23. This graph shows the performance of the four target systems on writing pixels of varying image sizes to the framebuffer in immediate, 0.5x zoom, and RGBA mode.

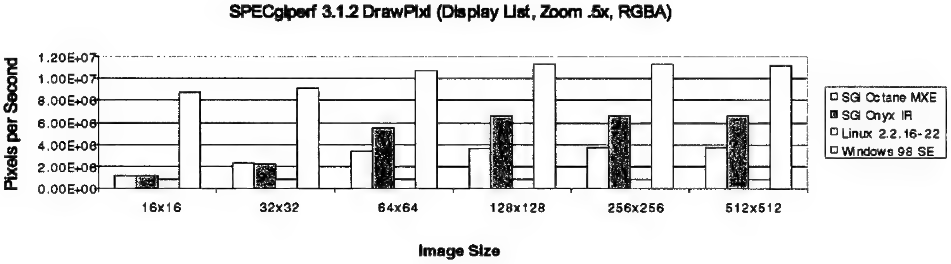


Figure 24. This graph shows the performance of the four target systems on writing pixels of varying image sizes to the framebuffer in display-list, 0.5x zoom, and RGBA mode.

4.1.4 ReadPixl (glreadPixels)

This script, as depicted in figures 25 and 26, tests how fast a rectangular array of pixels can be read from the framebuffer and stored in processor memory.

Once again, the SGIs are significantly faster in reading large images from the framebuffer. The SGI Octane is again faster than the Onyx and the GeForce is faster than the SGIs in reading images of sizes 64x64 or smaller.

4.1.5 Fillrate

The fill rate, as depicted in figure 27, is a measure of the speed at which primitives are converted to fragments and drawn into the framebuffer. Fragments are pixels in the framebuffer with color, alpha, depth and other data (not just the raw color data that appears in an image). Fill rates reflect the performance in the rasterization phase of a graphics pipeline and are reported as the number of pixels drawn per second.

Figure 25. This graph shows the performance of the four target systems on reading pixels of varying image sizes in RGB mode from the framebuffer.

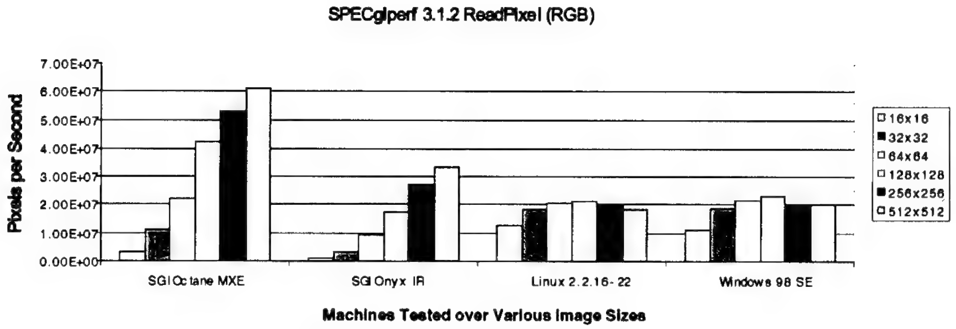


Figure 26. This graph shows the performance of the four target systems on reading pixels of varying image sizes in RGBA mode from the framebuffer.

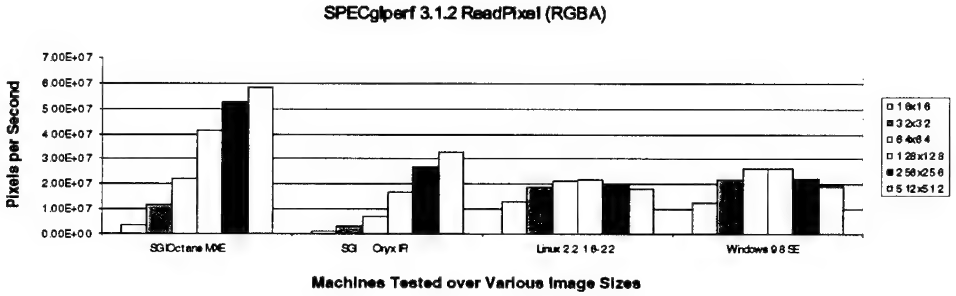
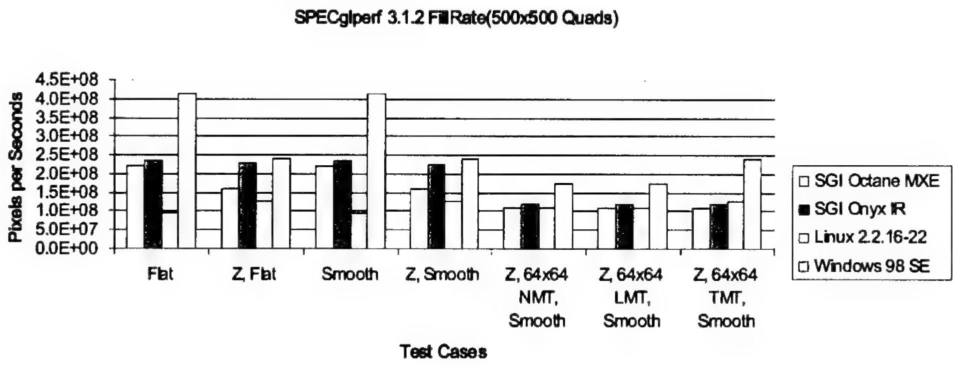


Figure 27. This graph shows the fill rate of each target system in various modes.



The Windows GeForce 256 is clearly the fastest here. Note that its greatest leads come when Z buffer is turned off. This indicates that the Z buffer implementation on the GeForce 256 is slower than that of the SGI's. In other words, the SGIs may have an edge on rendering scenes with high depth complexity.

4.1.6 LineFill

This script measures the effect of increasing primitive size on the drawing rates of line segments. Six graphs depicted in figures 28 through 33 are generated from the data collected.

The first two graphs (figures 28 and 29) show that the Windows/GeForce 256 can render more lines per second than the SGIs can. However, when the Z buffer is turned on, the drawing rate on the GeForce 256 drops sharply. The SGI Onyx soon outperforms the others when lines with pixel sizes of 3 or greater are drawn. Once again, this reveals that the Z buffer hardware on the GeForce 256 is slower than that of the SGIs'. With both the Z buffer and anti-aliasing turned on, the SGI Onyx leads from the start. Clearly we see that the SGI has an edge on its Z buffer and line antialiasing hardware implementation.

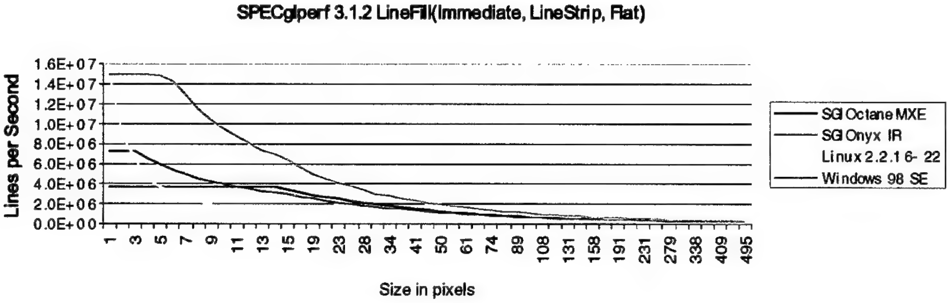


Figure 28. This graph shows the performance of the four target systems on rendering varying sizes of line strips in immediate, and flat shaded mode.

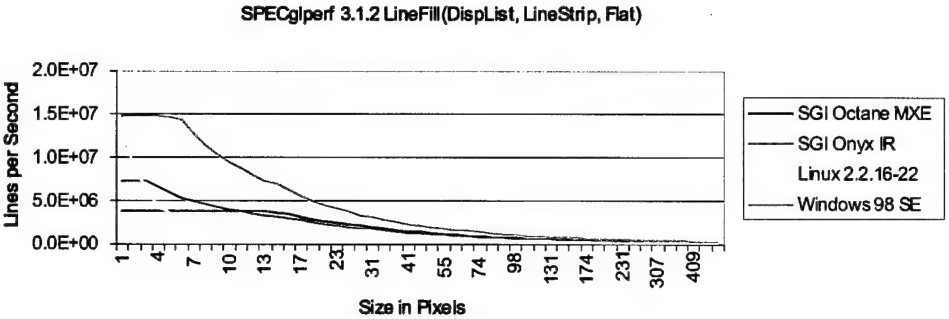


Figure 29. This graph shows the performance of the four target systems on rendering varying sizes of line strips in display-list, and flat shaded mode.

Figure 30. This graph shows the performance of the four target systems on rendering varying sizes of line strips in immediate, and flat shaded mode with the Z buffer turned on.

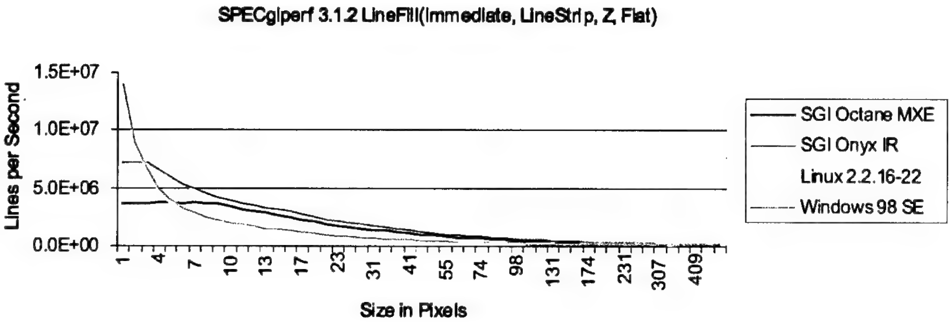


Figure 31. This graph shows the performance of the four target systems on rendering varying sizes of line strips in display-list, and flat shaded mode with the Z buffer turned on.

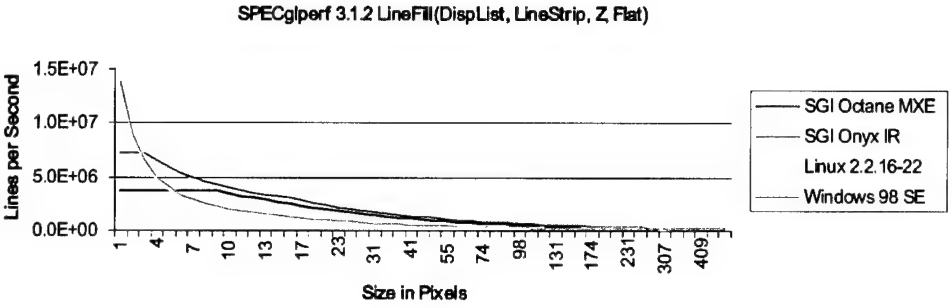


Figure 32. This graph shows the performance of the four target systems on rendering varying sizes of line strips in immediate, and flat shaded mode with both the Z buffer and antializsing turned on.

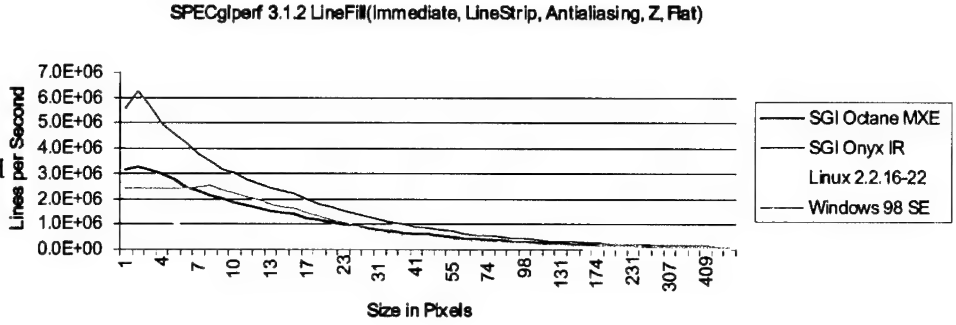
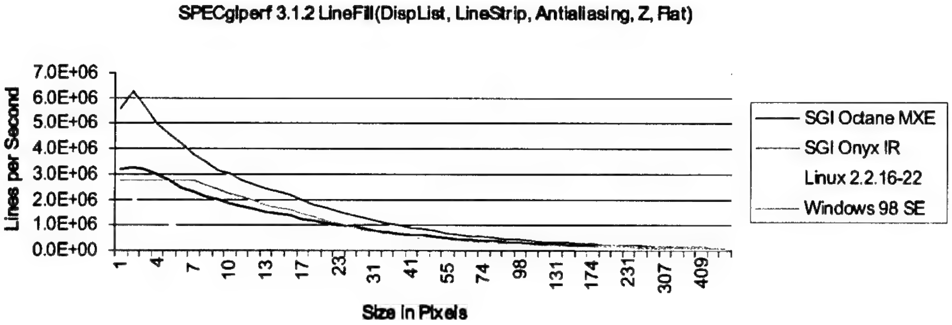


Figure 33. This graph shows the performance of the four target systems on rendering varying sizes of line strips in display-list, and flat shaded mode with the Z buffer turned on.



4.1.7 TriFill

Rather than measuring line strips, this test measures the effect of increasing primitive size on the drawing rates of triangle strips. Data from eight tests are collected and graphed (see figures 34 through 41).

The Windows/GeForce 256 consistently renders faster in this test. Comparing figures 34 and 36 figures 38 and 40 reveals that the GeForce 256 performance drops significantly when the Z buffer is turned on, whereas the performance the SGIs does not. This is consistent with earlier results that SGIs have a faster Z buffer implementation. No significant difference between the SGI and GeForce in either flat or smooth shading or between immediate and display-list mode was evidence.

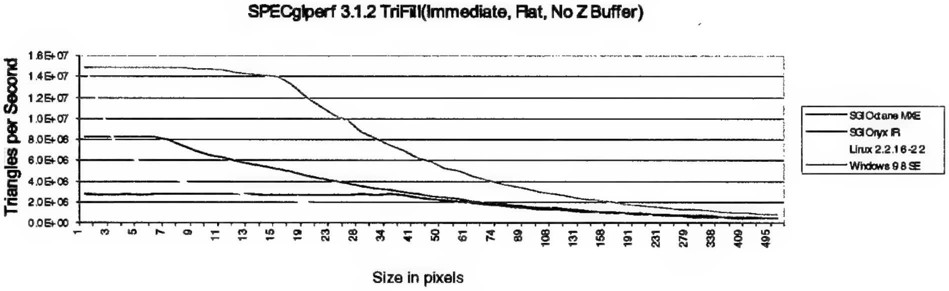


Figure 34. This graphs shows the performance of target systems on rendering triangles strips of varying sizes in immediate and flat shaded mode.

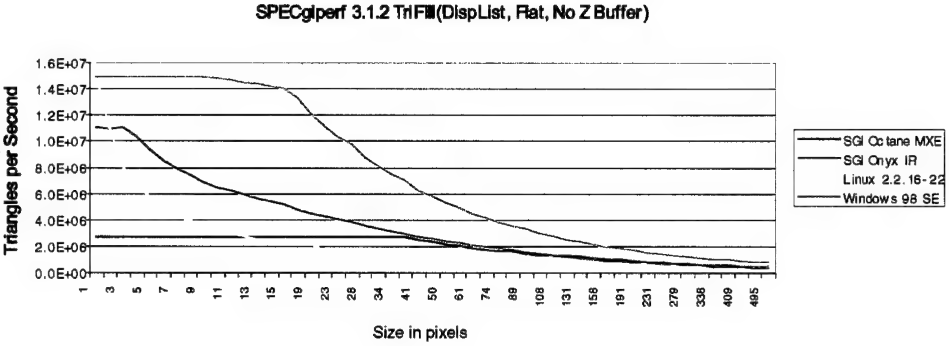


Figure 35. This graphs shows the performance of target systems on rendering triangles strips of varying sizes in display-list and flat shaded mode.

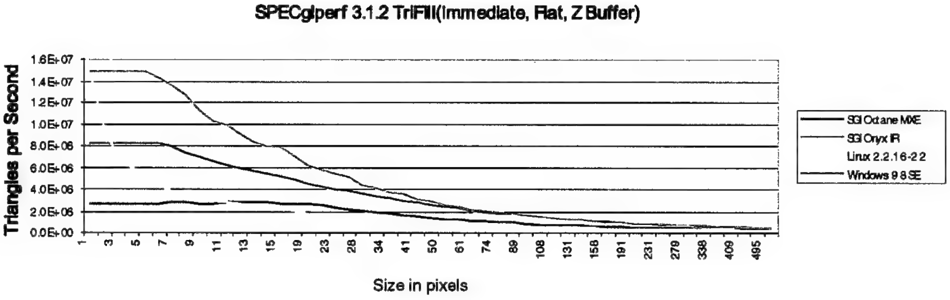


Figure 36. This graphs shows the performance of target systems on rendering triangles strips of varying sizes in immediate and flat shaded mode with Z buffer turned on.

Figure 37. This graphs shows the performance of target systems on rendering triangles strips of varying sizes in display-list and flat shaded mode with Z buffer turned on.

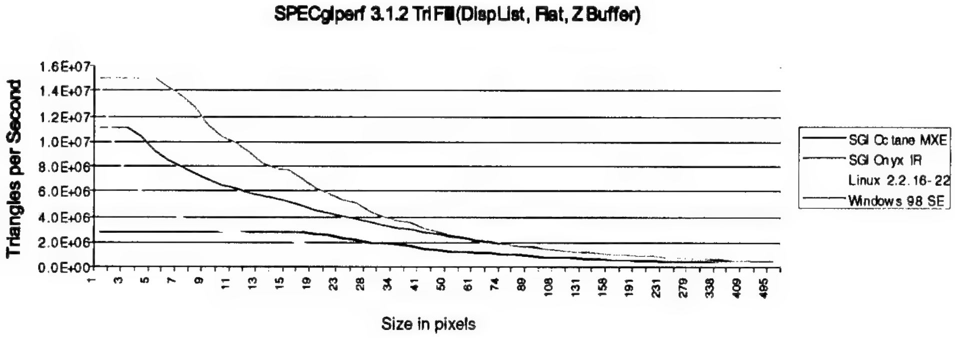


Figure 38. This graphs shows the performance of target systems on rendering triangles strips of varying sizes in immediate and smooth shaded mode.

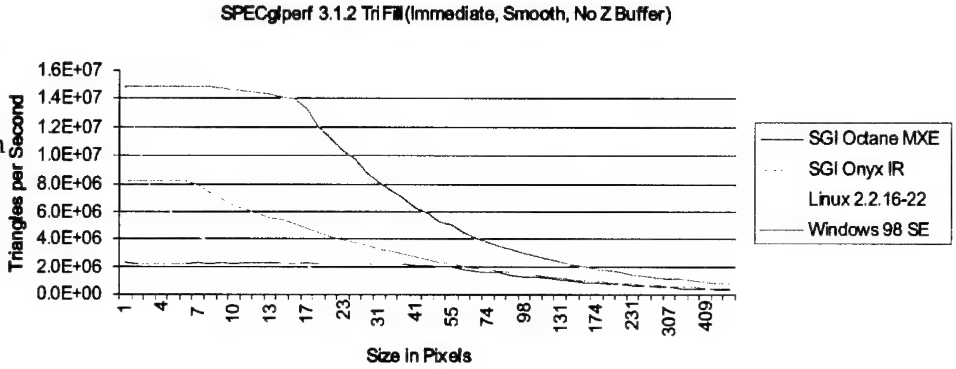
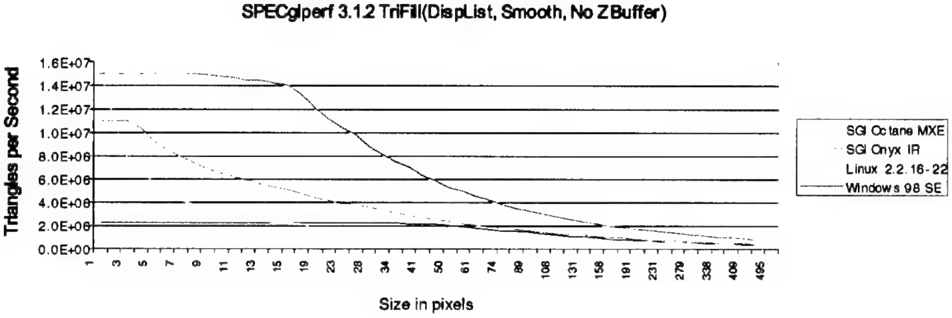


Figure 39. This graphs shows the performance of target systems on rendering triangles strips of varying sizes in display-list and smooth shaded mode.



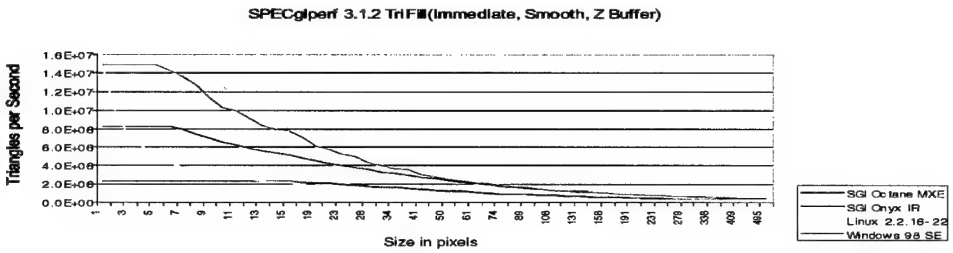


Figure 40. This graphs shows the performance of target systems on the rendering triangles strips of varying sizes in immediate and smooth shaded mode with Z buffer turned on.

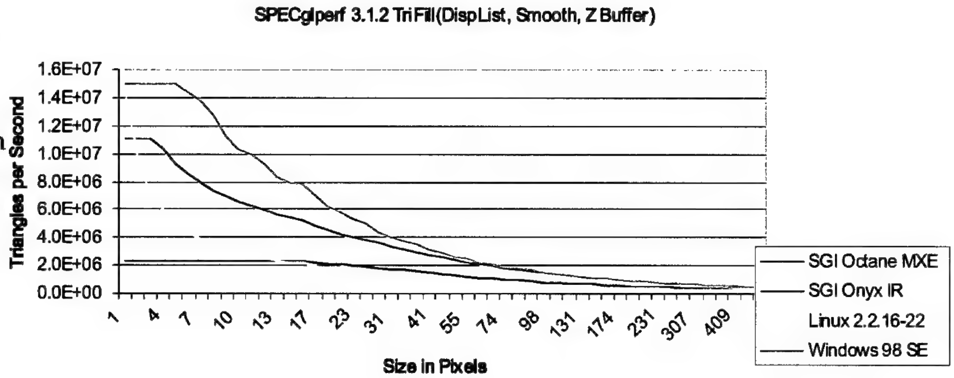


Figure 41. This graphs shows the performance of target systems on rendering triangles strips of varying sizes in display-list and smooth shaded mode with Z buffer turned on.

4.1.8 Light

This test, shown in figures 42 through 45, measures the effect of varying the number of enabled light sources on the drawing of triangle strips and quads primitives.

The graphs show that the GeForce 256 is the clear winner in rendering both types of primitives under various numbers of infinite light sources. However, the performance differences narrow between the Onyx and GeForce as more lights are added in the scene.

Figure 42. This graph shows the performance of the four target systems on rendering triangle strips with varying number of light sources in immediate and smooth-shaded mode with Z buffer turned on.

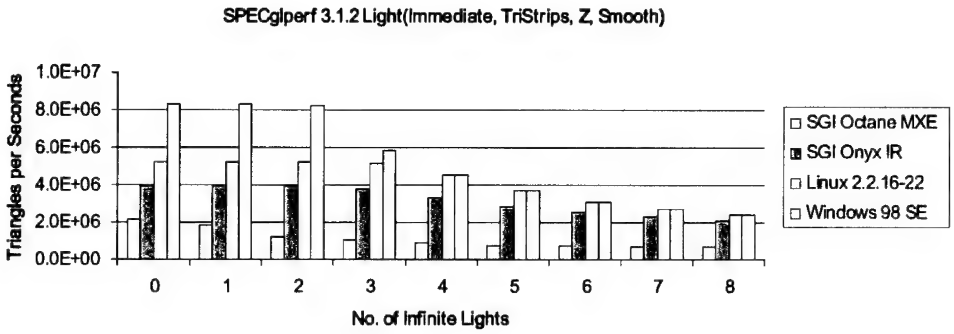


Figure 43. This graph shows the performance of the four target systems on rendering triangle strips with varying number of light sources in display-list and smooth-shaded mode with Z buffer turned on.

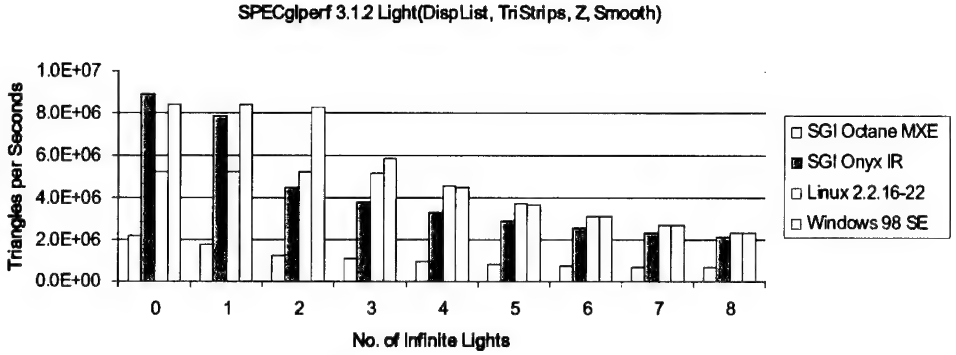


Figure 44. This graph shows the performance of the four target systems on rendering quads with varying number of light sources in immediate and smooth-shaded mode with Z buffer turned on.

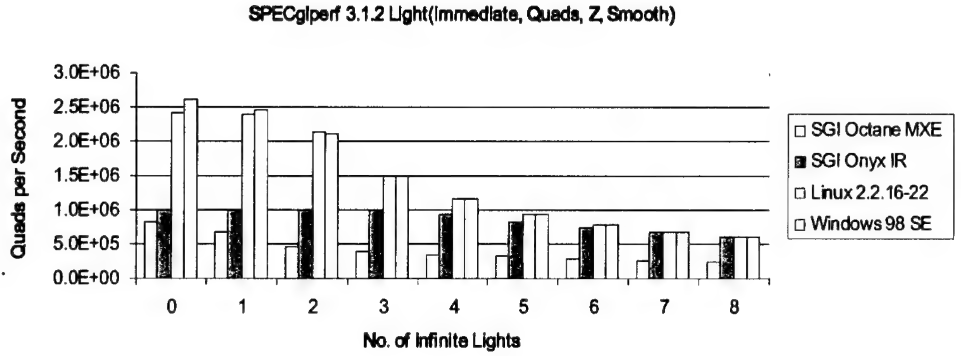
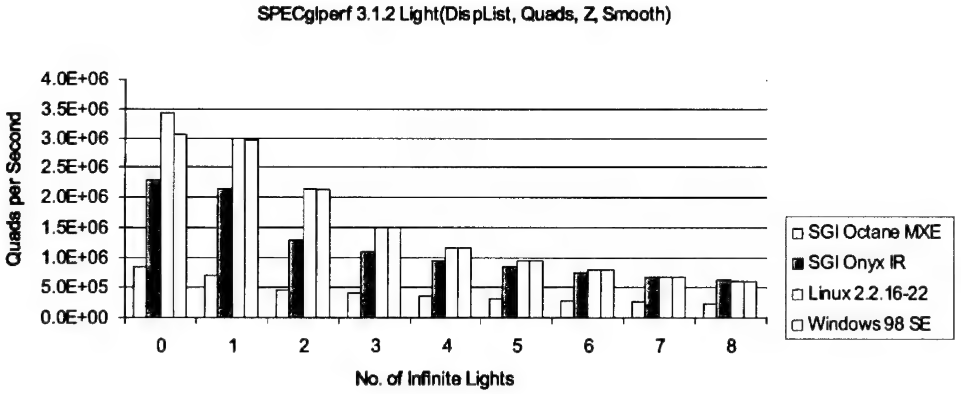


Figure 45. This graph shows the performance of the four target systems on rendering quads with varying number of light sources in display-list and smooth-shaded mode with Z buffer turned on.



4.1.9 OPClist

The OPClist scripts contain a number of tests for a variety of graphics primitives and other operations (such as window-clears). These tests are the closest parallel to primitive-level results available from most vendors today. Seven results are presented in figures 46 through 56.

The graph in figure 46 reveals that the SGI Onyx is faster in clearing the color buffer than the GeForce 256, but slower in clearing the depth buffer. This is also one of the few cases in which the Linux system performs better than its Windows counterpart. Observing the rest of the results, one sees that the Win98/GeForce 256 outperforms the SGI Onyx in all disjoint primitives tests that include points, lines, triangles, and quads. One also sees that, once again, the Linux/GeForce 256 performed significantly poorer than its Windows counterpart in all the primitives' tests.

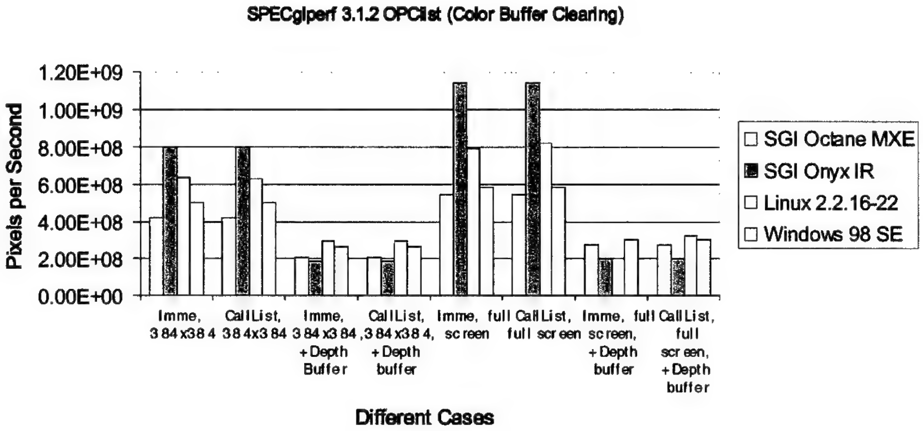


Figure 46. This graph shows the performance of the four target systems on clearing the color buffer in various modes.

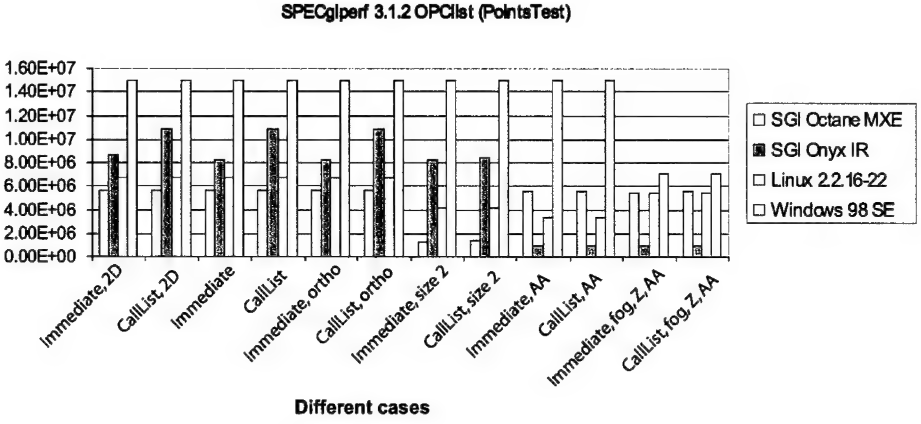


Figure 47. This graph shows the performance of the four target systems on the rendering of points with various modes turned on.

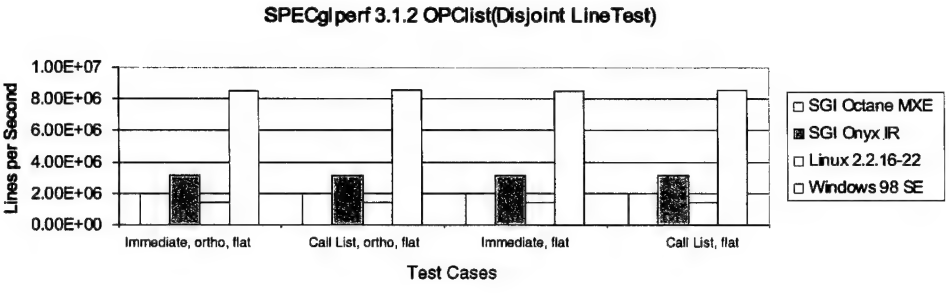


Figure 48. This graph shows the performance of the four target systems on rendering disjoint lines in various modes.

Figure 49. This graph shows the performance of the four target systems on rendering disjoint triangles in various modes.

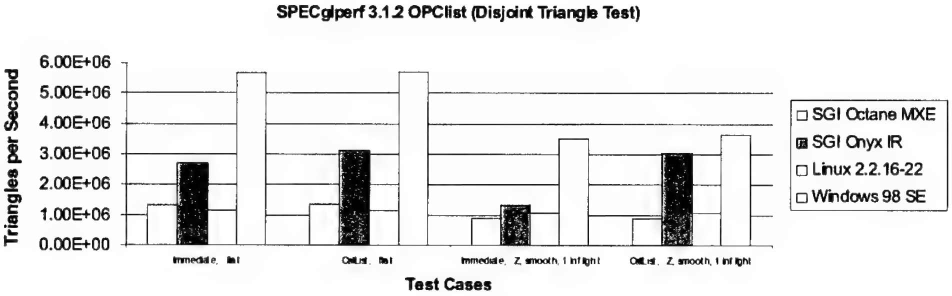


Figure 50. This is one of the five graphs that show the performance of the four target systems on rendering disjoint quads in various modes.

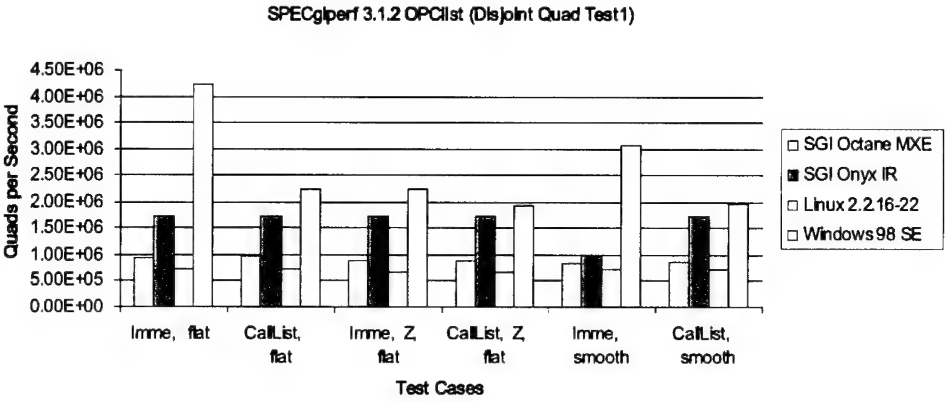


Figure 51. This is second of the five graphs that show the performance of the four target systems on rendering disjoint quads in various modes.

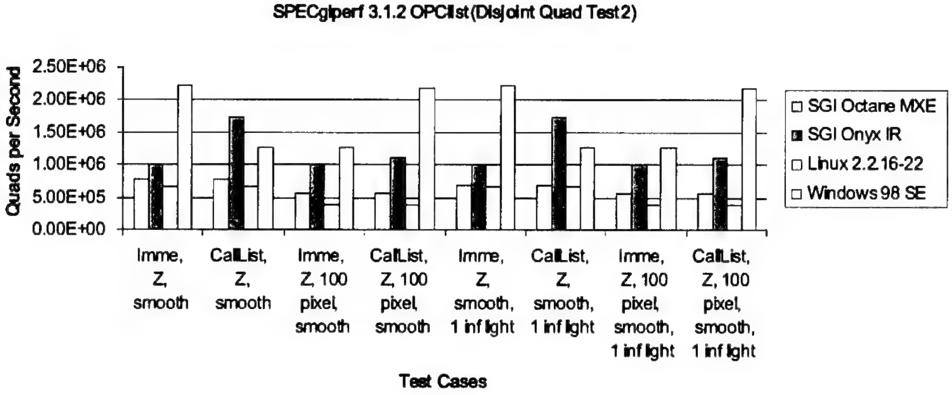


Figure 52. This is the third of the five graphs that show the performance of the four target systems on rendering disjoint quads in various modes.

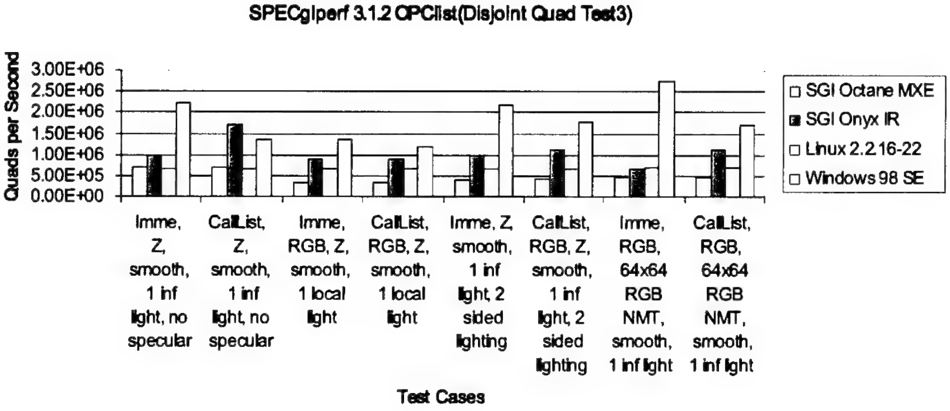


Figure 53. This is forth of the five graphs that show the performance of the four target systems on rendering disjoint quads in various modes.

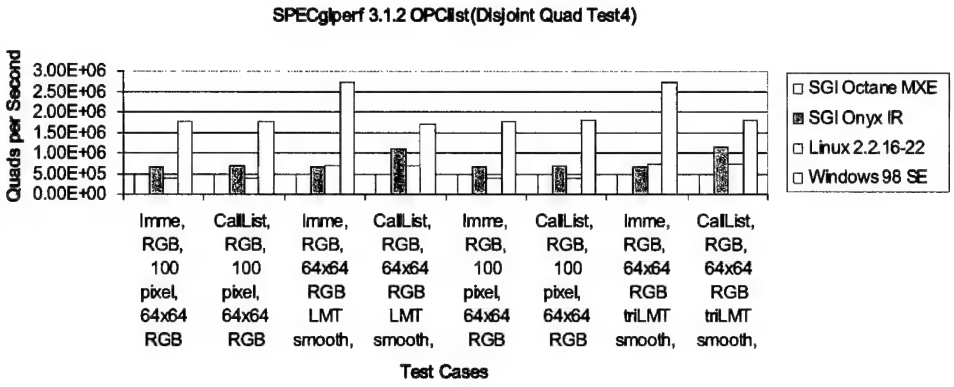


Figure 54. This is last of the five graphs that show the performance of the four target systems on rendering disjoint quads in various modes.

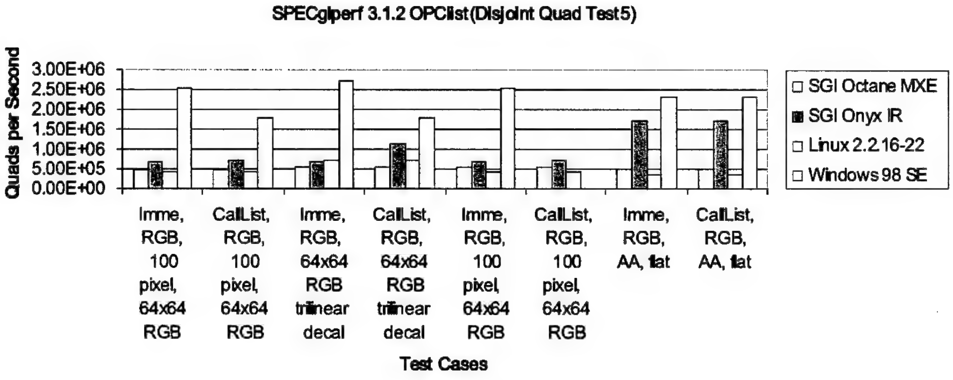


Figure 55. This graph shows the performance of the four target systems on the rendering of 10-sided disjoint polygons in various modes.

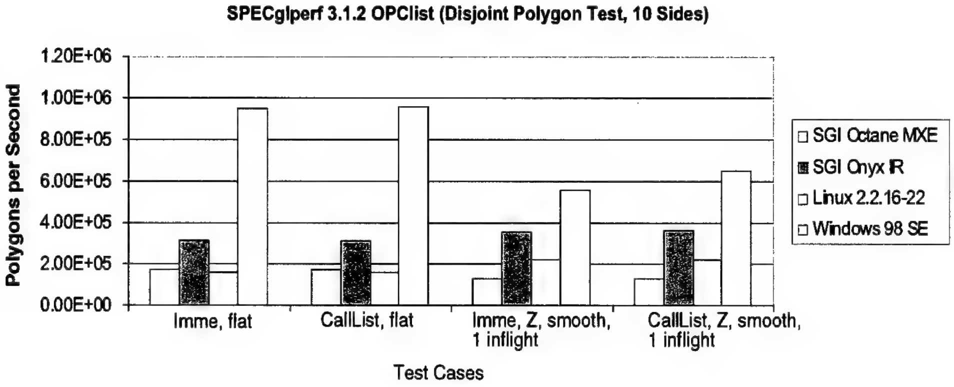
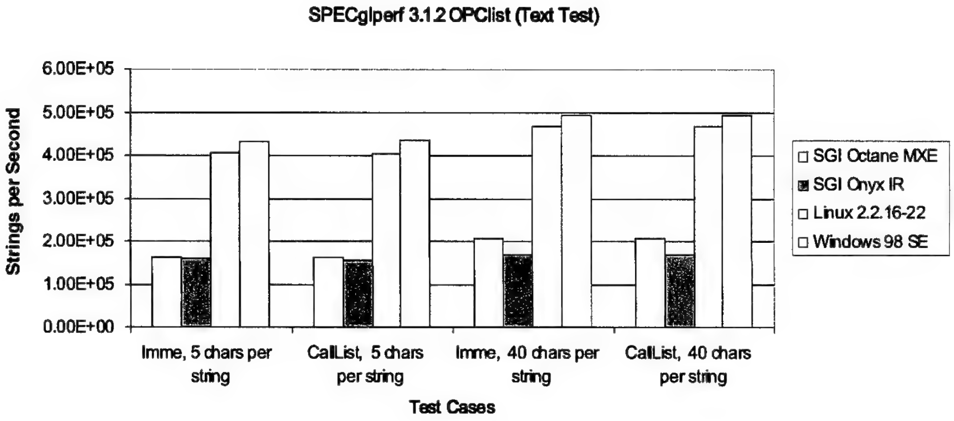


Figure 56. This graph shows the performance of the four target systems on rendering text strings in various modes.



4.1.10 TexImage

This script, shown in figures 57 through figure 62, tests how fast the graphics hardware can draw textures with increasing image sizes. The results are in texels per second. Figures 61 and 62 show how fast textures can be bound with the use of either glCallList or texture object with or without mipmapped textures.

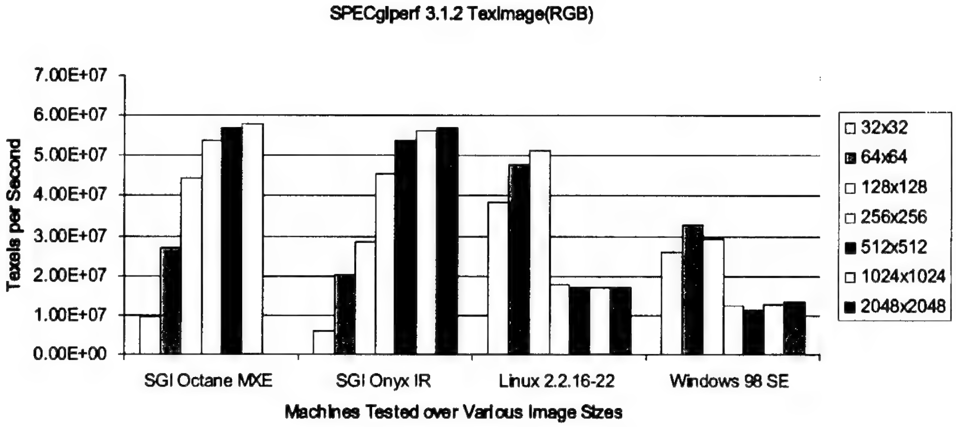


Figure 57. This graph shows the performance of the four target systems on rendering images of various sizes in RGB format.

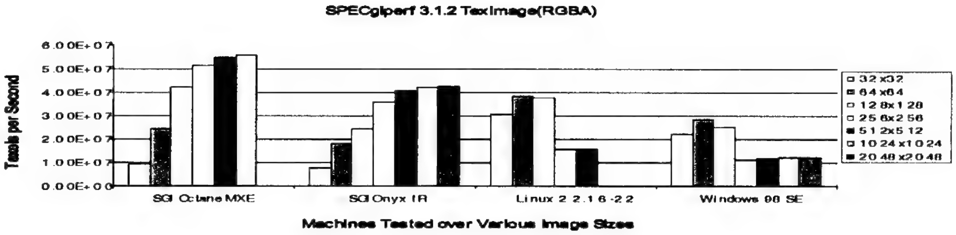


Figure 58. This graph shows the performance of the four target systems on rendering images of various sizes in RGBA format.

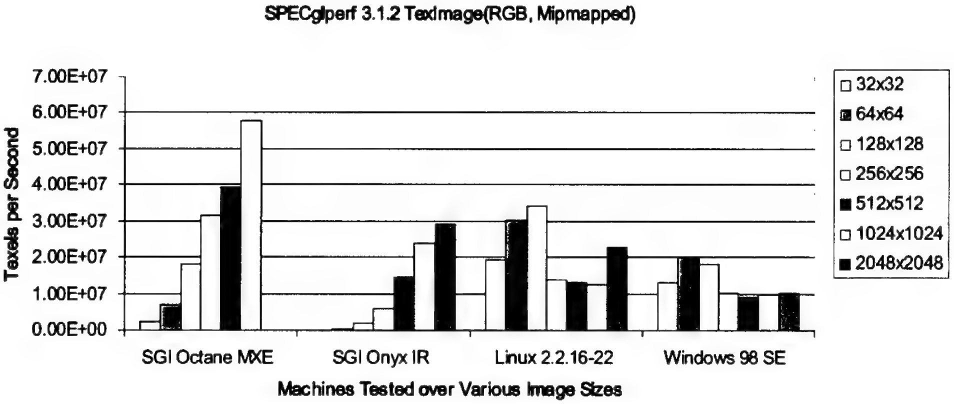


Figure 59. This graph shows the performance of the four target systems on rendering mipmapped textures of various sizes in RGB format.

The graphs in figures 57 through 60 clearly show that the SGIs are faster in rendering textures with images greater than 128x128. They also show that the Octane is faster than the Onyx; the Linux/GeForce 256 is faster than its Windows counterpart in this respect. Figures 61 and 62 reveal that the GeForce 256 is faster in binding textures whether it is mipmapped or not.

Figure 60. This graph shows the performance of the four target systems on rendering mipmapped textures of various sizes in RGBA format.

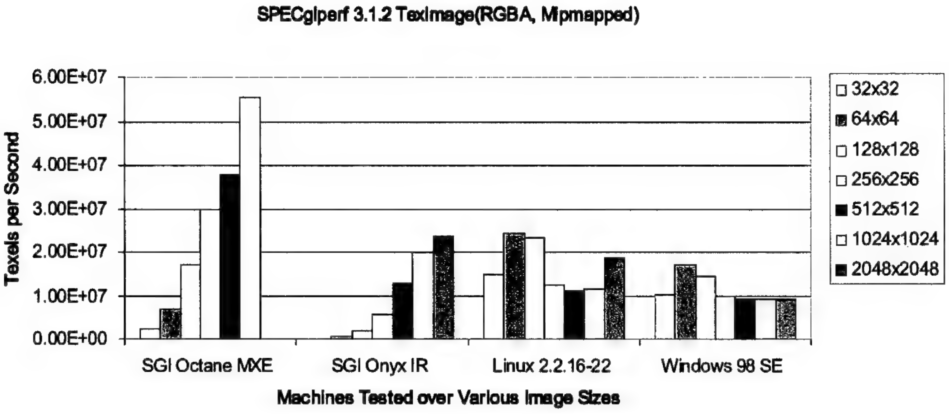


Figure 61. This graph shows the performance of the four target systems on bounding non-mipmapped textures.

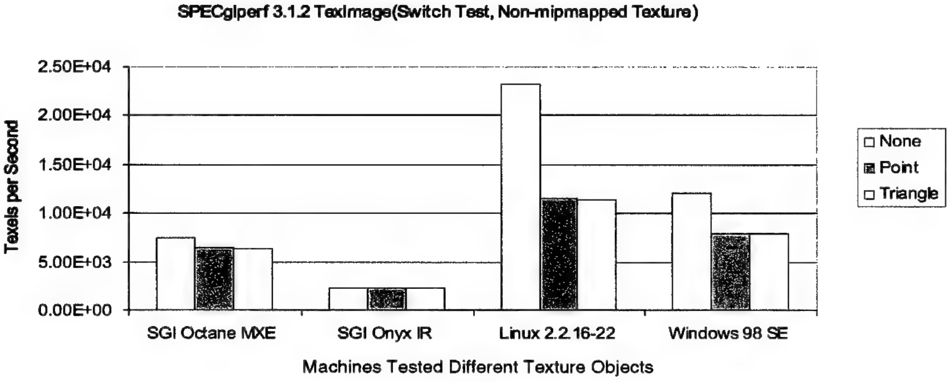
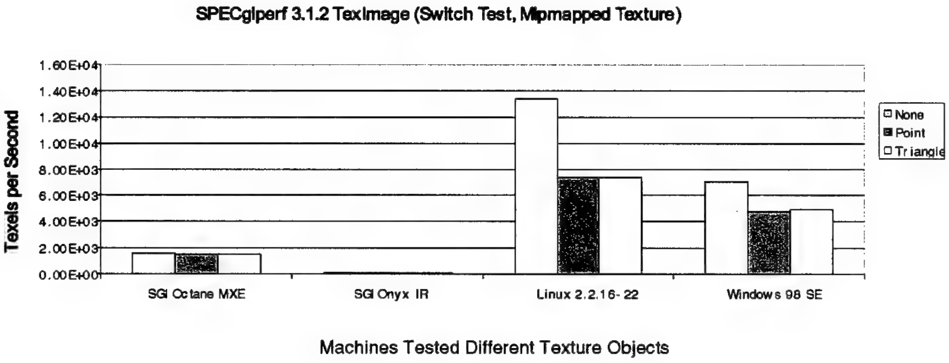


Figure 62.This graph shows the performance of the four target systems on bounding mipmapped textures.



4.2 SPECglperf Conclusion

Most graphics hardware has a set of fast paths that execute a subset of rendering operations much faster than others. The rendering operations performed on these fast paths are based on the primitives and modes directly supported by the underlying hardware. The use of a primitive type or rendering mode that is not directly supported by the hardware causes the graphics-rendering pipeline to fall back to a less optimal path or to software rendering. If one knows the hardware fast paths of a particular system, one can design an application that will stay on them to achieve best performance.

In this benchmark, the GeForce 256 is significantly faster than the SGIs in rendering primitives whether they are batched or not, smooth- or flat-shaded. On the other hand, the SGIs are faster in their Z buffer, antialiasing and large-texture rendering. The Onyx is faster than the Octane in most cases except in texture rendering where the Octane is consistently faster. In regards to NVIDIA's 0.95 Xfree 4 Linux driver, the benchmark results revealed that there is obviously room for improvement. All in all, the GeForce 256 performed surprisingly well against the SGIs. NVIDIA's latest effort to boost OpenGL performance for Linux users is a successful one.

5. Summary

5.1 Performance Per Dollar

One might assume that since the Micron PC outperforms the Octane (up to five times faster in some cases), it must cost much more. Contrarily, the Micron PC costs about $1400 as compared to the over $10,000 price tag for the Octane. The next question might be "Will the better performance of the Octane2 be great enough to justify its $14000 CPU/graphics upgrade cost?" Until we do the upgrade and perform the same benchmark, the question cannot be answered. Nevertheless, according to SGI, "Octane2 equipped with single or dual MIPS R12000A 400 MHz processors offers three times the graphics price/performance of Octane1 and boasts 33% faster CPU performance." [7] These claims remain to be proven. However, the new Octane2 does have some advantages over an Intel-based PC. Two of these are: Octane's V8 graphics has 128 MB graphics memory including up to 104 MB texture memory as opposed to 64 MB on a GeForce or Quadro chip-based board and Octane2's system board can accommodate up to 8 GB of system memory as compared to 2 GB on the best PC system board.

On the other hand, the GeForce 256 used in this test was one year old or two generations old! It was released in August of 1999. Since then, NVIDIA released the GeForce2 GTS in May of 2000, the Quadro2 Pro in July, the GeForce2 Ultra in August of 2000 and GeForce3 in March of 2001. Currently, the GeForce2 GTS sells for about $200, the GeForce2 Ultra sells for about $300, the GeForce3 sells for about $500, and the Quadro2 Pro about $1000. On the other hand, SGI's entire line of Intel processor-based (NT/Linux) visual workstations also uses a "custom performance enhanced" Quadro GPU from NVIDIA [8]. As far as bang for the buck goes, the GeForce is, without a doubt, the winner.

5.2 The Future

With the rapid development of PC-based graphics cards and the maturing of Linux, the performance gap between PC and SGI/SUN-based workstations is narrowing. Recently, NVIDIA released the first mobile graphics processing unit (GPU), GeForce2 Go, for laptops. This is a step toward being able to render complicated 3D applications, e.g., VGIS, on a laptop. On the other hand, SGI has been a recognized leader in 3D graphics for the past 15 years, but changes are evident. Looking at the benchmark scores, one can see that SGI is losing ground on its low- to middle-range line of products. SGI will have to narrow its performance per dollar gap with the intel-based systems in the future to justify its higher prices. Benchmarking our newly acquired SGI Origin 3200, Octane2 and NVIDIA's GeForce3 would further show how state-of-art graphics hardware compares.

References

1. P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust and G. Turner (1997), "An Integrated Global GIS and Visual Simulation System", Submitted to IEEE Trans. Visualization and Computer Graphics.
2. M. Woo, J. Neider, and T. Davis (1997), "OpenGL Programming Guide", Addison-Wesley Developers Press.
3. SPECviewperf's URL: http://www.spec.org/gpc/opc.static/vp50.htm
4. SPECglperf's URL: http://www.spec.org/gpc/opc.static/glperf.htm
5. OpenGL Performance Characterization (OPC) organization's URL: http://www.spec.org/gpc/opc.static/overview.htm
6. PNG's URL: http://www.libpng.org/pub/png/
7. SGI news release, June 26, 2000, URL: http://www.sgi.com/newsroom/press_releases/2000/june/octane2.html
8. NVIDIA news release, May 18, 2000, URL: http://www.NVIDIA.com/News/Pages.nsf/5b9ac8fa8e1ed0448825685c0066b465/8875134553d547e1882568e80017b5b7

Distribution

Admnstr
Defns Techl Info Ctr
ATTN DTIC-OCP
8725 John J Kingman Rd Ste 0944
FT Belvoir VA 22060-6218

DARPA
ATTN S Welby
3701 N Fairfax Dr
Arlington VA 22203-1714

Ofc of the Secy of Defns
ATTN ODDRE (R&A T)
The Pentagon
Washington DC 20301-3080

AMCOM MRDEC
ATTN AMSMI-RD W C McCorkle
Redstone Arsenal AL 35898-5240

US Army TRADOC
Battle Lab Integration & Techl Dirctrt
ATTN ATCD-B
ATTN ATCD-B J A Klevecz
FT Monroe VA 23651-5850

ElanTech, Inc.
4105 Nesconset Dr
Bowie MD 20716

Dir for MANPRINT
Ofc of the Deputy Chief of Staff for Prsnnl
ATTN J Hiller
The Pentagon Rm 2C733
Washington DC 20301-0300

SMC/CZA
2435 Vela Way Ste 1613
El Segundo CA 90245-5500
TECOM
ATTN AMSTE-CL
Aberdeen Proving Ground MD 21005-5057

US Army ARDEC
ATTN AMSTA-AR-TD
Bldg 1
Picatinny Arsenal NJ 07806-5000

US Army Info Sys Engrg Cmnd
ATTN AMSEL-IE-TD F Jenia
FT Huachuca AZ 85613-5300

US Army Natick RDEC Acting Techl Dir
ATTN SBCN-T P Brandler
Natick MA 01760-5002

US Army Simulation Train & Instrmntn
Cmnd
ATTN AMSTI-CG M Macedonia
ATTN J Stahl
12350 Research Parkway
Orlando FL 32826-3726

US Army Tank-Automtv Cmnd RDEC
ATTN AMSTA-TR J Chapin
Warren MI 48397-5000

Nav Surfc Warfare Ctr
ATTN Code B07 J Pennella
17320 Dahlgren Rd Bldg 1470 Rm 1101
Dahlgren VA 22448-5100

Hicks & Assoc Inc
ATTN G Singley III
1710 Goodrich Dr Ste 1300
McLean VA 22102

US Army Rsrch Lab
ATTN AMSRL-CI-CT R C Kaste
Aberdeen Proving Ground MD 21005

Director
US Army Rsrch Lab
ATTN AMSRL-RO-D JCI Chang
ATTN AMSRL-RO-EN W D Bach
PO Box 12211
Research Triangle Park NC 27709

Distribution (cont'd)

US ARDEC
ATTN AMSRL-CI-C J Gowens
ATTN AMSRL-CI-CB C Winslow
ATTN AMSRL-CI-CB J DeHart
ATTN AMSRL-CI-CB J Gurney
ATTN AMSRL-CI-CB K Deacon
ATTN AMSRL-CI-CB L Tokarcik
ATTN AMSRL-CI-CB R Meyers
ATTN AMSRL-CI-CB R Winkler
ATTN AMSRL-CI-CB S Ho (5 copies)
ATTN AMSRL-CI-CB H Nguyen
ATTN AMSRL-CI-IS-R Mail & Records Mgmt
ATTN AMSRL-CI-IS-T Techl Pub (2 copies)
ATTN AMSRL-CI-OK-TL Techl Lib (2 copies)
ATTN AMSRL-DD J M Miller

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE November 2001

3. REPORT TYPE AND DATES COVERED Final,

4. TITLE AND SUBTITLE OpenGL Performance Evaluation on Multiple Computer Platforms

5. FUNDING NUMBERS
DA PR: N/A
PE: 61102A

6. AUTHOR(S) Sean Ho

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
Attn: AMSRL-CI-CB email: sean@mail.army.mil
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-2374

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
2800 Powder Mill Road
Adelphi, MD 20783-1197

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
ARL PR: 1FE3F3
AMS code: 611102H4411

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

With the recent advent of 3D graphics hardware for personal computer (PC), it is worthwhile to exploit the cost effectiveness and OpenGL performance issues among currently available commercial off-the-self (COTS) computers. Graphics hardware vendors typically list several gross measurements of system performance when releasing new graphics hardware. Often these coarse or subjective figures do not represent how a software application performs. On the other hand, one seldom sees the same benchmark performed on machines across multiple platforms and operating systems, i.e., Intel-based PCs and RISC-based UNIX workstations. This document reports the results obtained from running two OpenGL benchmark programs, SPECviewperf 6.1.2 and SPECglperf 3.1.2, on existing computer workstations at ARL.

14. SUBJECT TERMS Benchmark, primitives, frames per second

15. NUMBER OF PAGES 35

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UL